Práctica 2

Introducción a las Redes Neuronales Artificiales

28 de Marzo de 2014

AUTORES

Mauricio Guerreiro Quatrin
Olympia Muruzábal Ishigetani

Índice

En esta práctica se nos pide implementar y analizar una red neuronal multicapa entrenada con el algoritmo de retropropagación de errores. Realizando cada una de las siguientes tareas:

Tarea 1: Implementación de la red neuronal

La primera tarea consiste en implementar de forma genérica el algoritmo de retropropagación de errores para entrenar una red con una sola capa oculta y funciones de activación sigmoide bipolar:

$$f(x) = \frac{2}{1 + \exp(-x)} - 1$$

$$f'(x) = \frac{1}{2}[1 + f(x)][1 - 1f(x)]$$

Hemos seguido el pseudocódigo proporcionado en las transparencias de teoría[1] para realizar la implementación del algoritmo de retropropagación, por lo tanto no comentaremos la codificación del algoritmo en este documento.

Cabe mencionar que partimos de las clases implementadas en la primera práctica, donde añadimos funcionalidades necesarias para esta práctica que no se tuvieron en cuenta en la práctica anterior. Concretamente en *InputData* e *InputRow* se han realizado las modificaciones necesarias para poder clasificar más de dos clases y se han añadido funciones para calcular la normalización de los datos.

La clase Backpropagation tiene las siguientes funciones:

- **setData**: prepara los datos de entrada con ayuda de ***InputData***.
- **initializeWeights**: inicializa una matriz de pesos y el sesgo con valores aleatorios entre [-0.5,0-5].
- **bipolarSigmoid**: devuelve el resultado de aplicar la función sigmoidal bipolar.
- **bipolarSigmoidPrima**: calcula la derivada de la función sigmoidal a partir del resultado de la función sigmoidal.
- **startTraining**: ejecuta el entrenamiento aplicando el algoritmo de retropropagación visto en teoría. Y donde la condición de parada se da si el error cuadrático medio es menor que un umbral dado (0.1) ó si la diferencia entre el error cuadrático de la época anterior y la actual es inferior a un umbral (0.000001), como mucho se realizarán 1000 épocas.
- **startTest**: ejecuta el test usando la red ya entrenada, escogiendo la clase predicha por la red correspondiente a la neurona con mayor activación.

[1] Neurocomputación Tema3 (pág. 10-13)

Tarea 2: chequeo del funcionamiento de la red con los ficheros de la práctica anterior

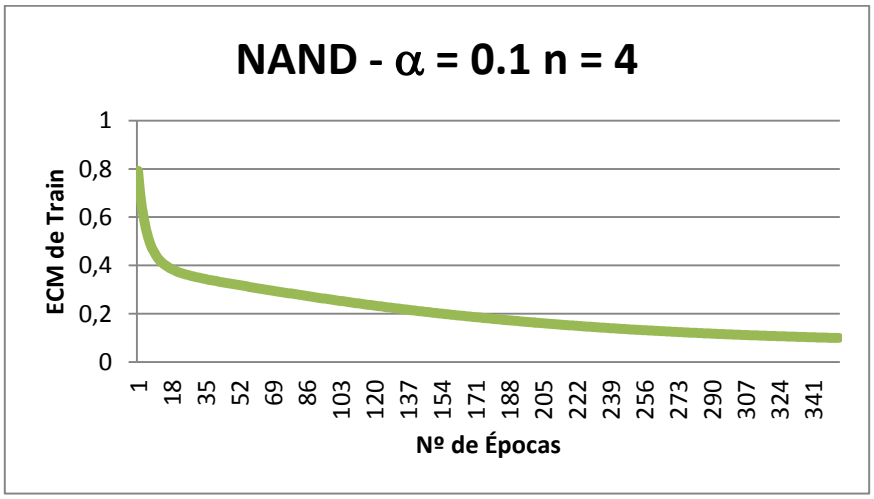
En este apartado probaremos el funcionamiento de la red neuronal multicapa implementada en el apartado anterior, utilizando para ello las mismas bases de datos de la práctica anterior. Las pruebas se han realizado variando el número de neuronas en la capa oculta y el valor de la constante de aprendizaje.

Resultados

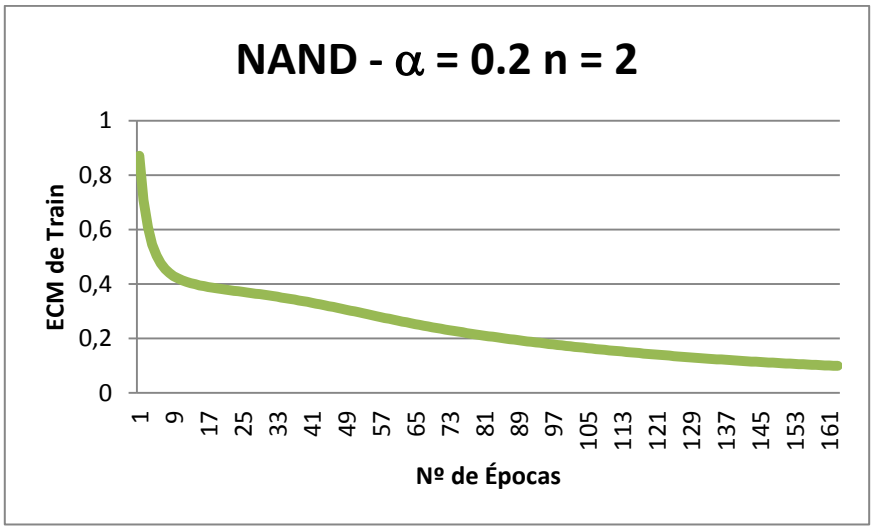
A continuación, se muestran los resultados obtenidos en tras realizar las pruebas. En el título de cada gráfica se indica el valor de la constante de aprendizaje (α) y el número de neuronas en la capa oculta (n). En las gráficas se representa el valor del error cuadrático medio (ECM) en cada época durante el entrenamiento. Sobre cada gráfica se indica la tasa de error en el test.

nand.txt

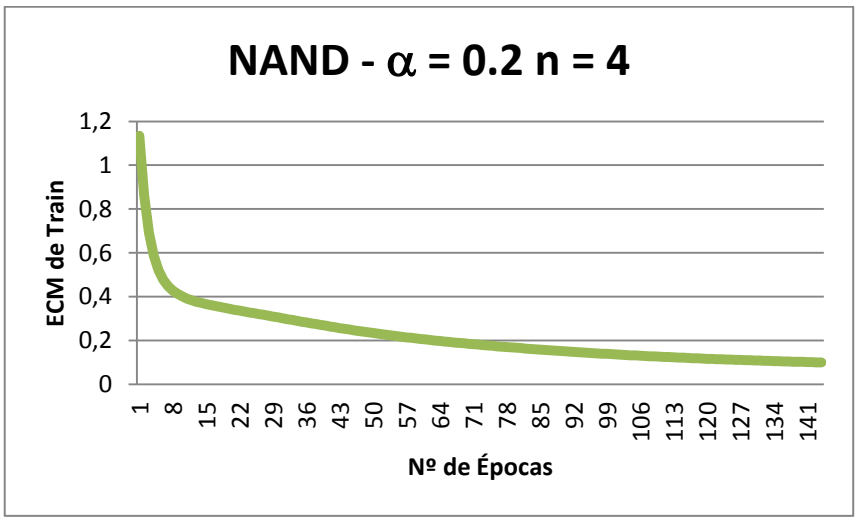
Tasa de error del Test: 0%



Tasa de error del Test: 0%

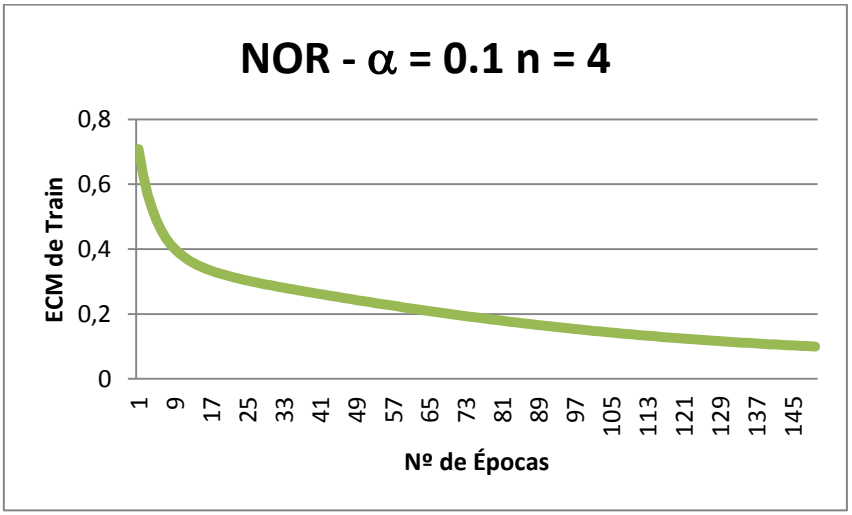


Tasa de error del Test: 0%

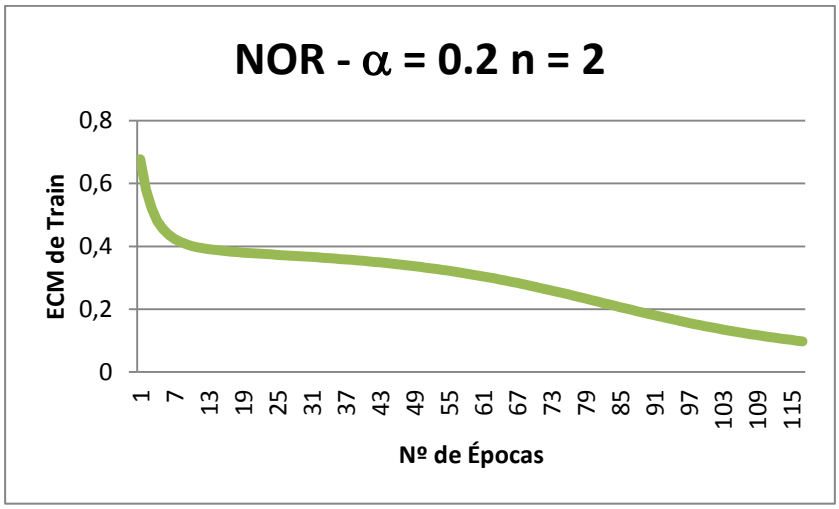


nor.txt

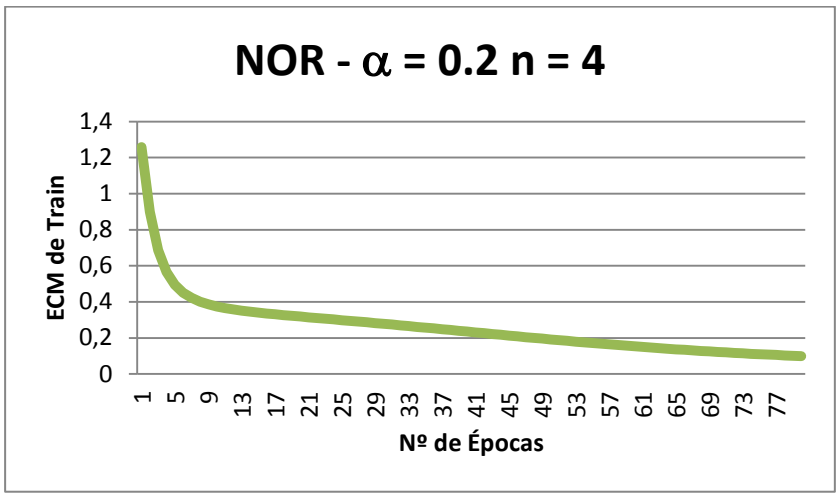
Tasa de error del Test: 0%



Tasa de error del Test: 0%

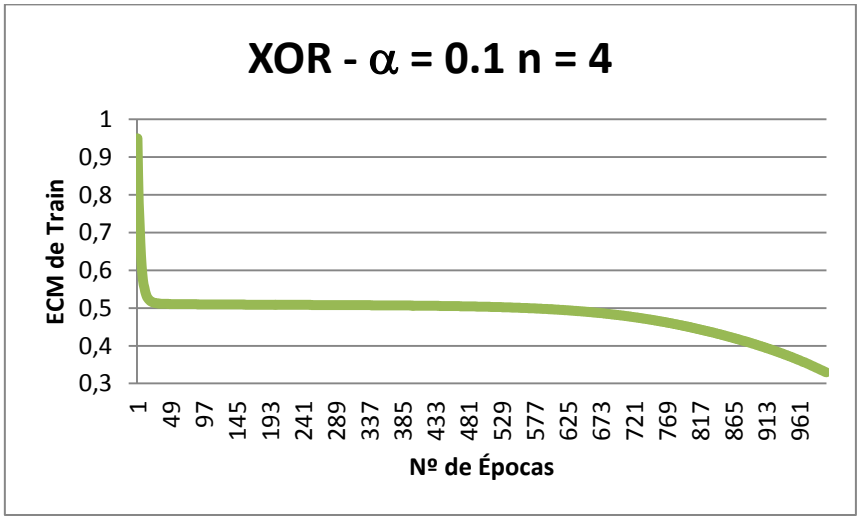


Tasa de error del Test: 0%

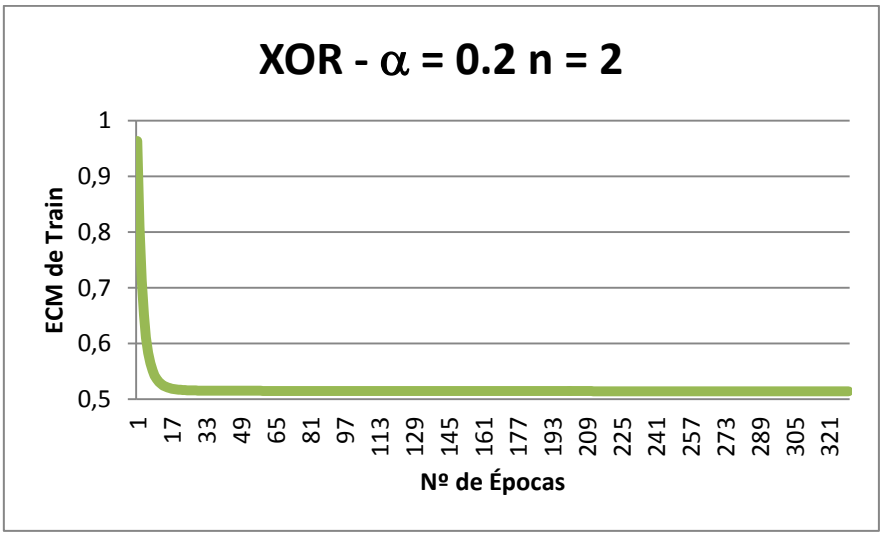


xor.txt

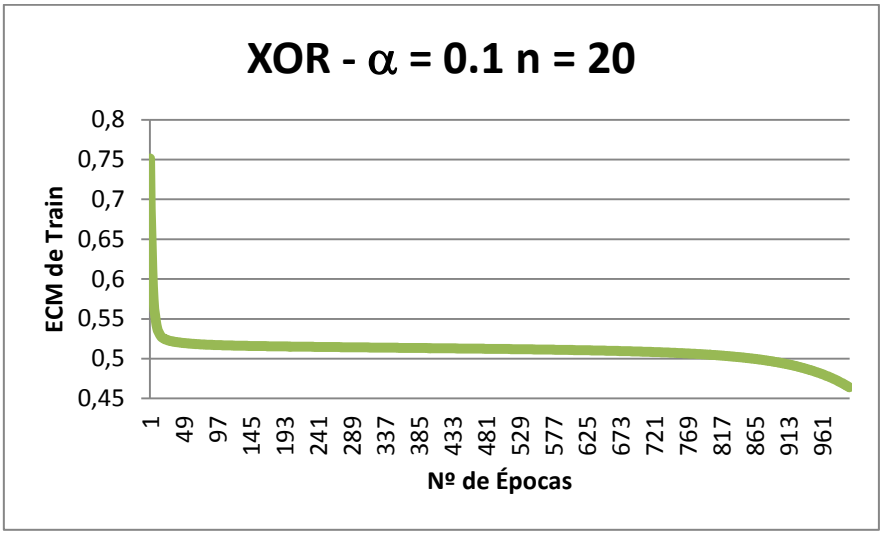
Tasa de error del Test: 50%



Tasa de error del Test: 25%

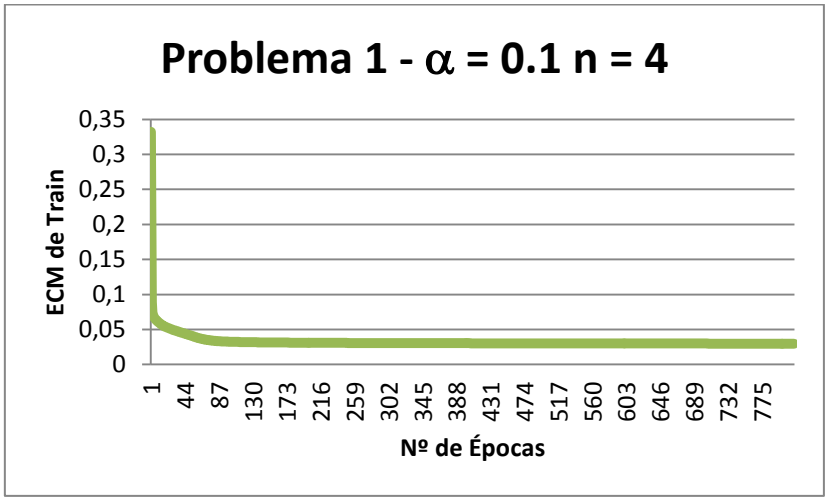


Tasa de error del Test: 0%

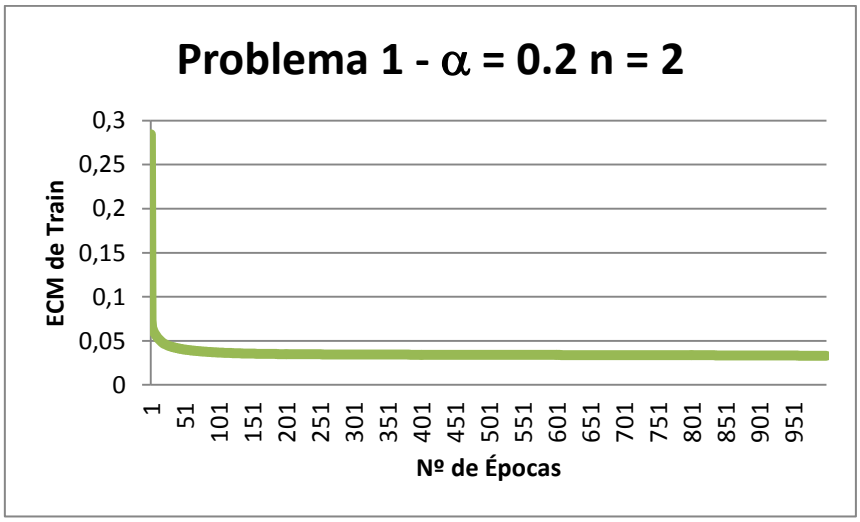


problema_real1.txt

Tasa de error del Test: 3.004291845493562%

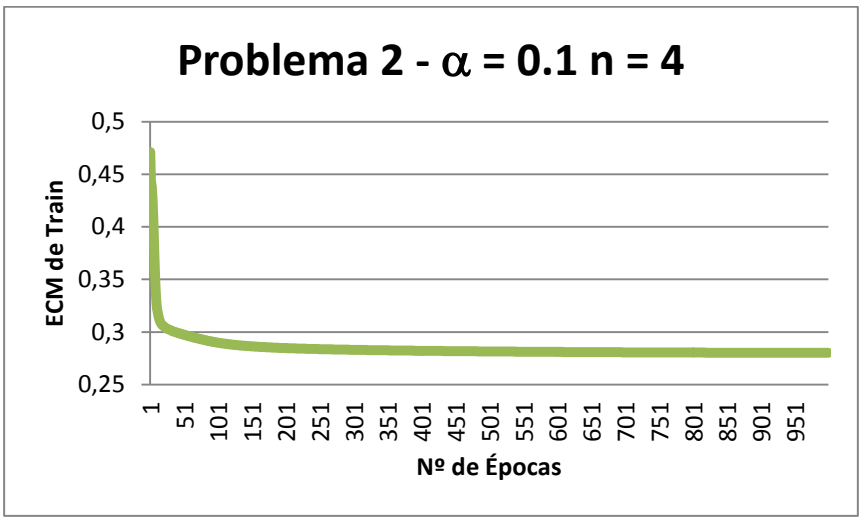


Tasa de error del Test: 3.004291845493562%

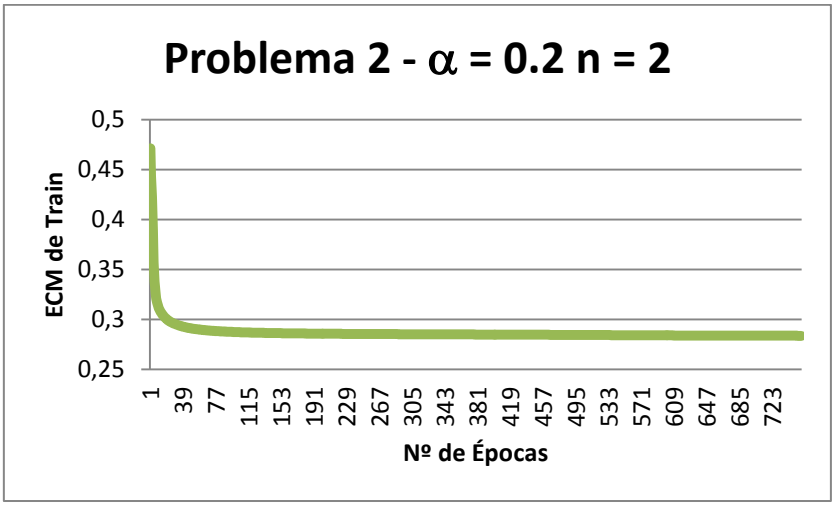


problema_real2.txt

Tasa de error del Test: 20.32085561497326%



Tasa de error del Test: 20.32085561497326%



Análisis de los resultados

En general podemos comentar que con una constante de aprendizaje menor son necesarios más ciclos para alcanzar el mínimo, debido a que el aprendizaje es más lento. A su vez, podemos observar que con un mayor número de neuronas en la capa oculta son necesarios menos ciclos. Por lo que optamos a realizar pruebas con una constante de aprendizaje del 0.1 y 0.2, con 4 neuronas en la capa oculta y 2 neuronas respectivamente.

A primera vista podemos observar que en todas las gráficas el error cuadrático medio convergen, salvo en el caso de la puerta lógica XOR, y la tasa de error en de los test es mínima, por lo que podemos decir que el algoritmo de propagación hacia atrás funciona.

Tal y como muestran las gráficas de las puertas lógicas NAND y NOR se alcanza el mínimo absoluto en pocos ciclos y la tasa de error de test es del 0%. A diferencia de la puerta XOR que requiere de muchas más épocas para clasificarlo correctamente. Realizamos varias pruebas para el XOR puesto que tendía quedarse en un mínimo local, y comprobamos que era necesaria una constante de aprendizaje muy pequeña y muchas más épocas para que alcanzara el mínimo absoluto.

En el caso de los problemas reales podemos observar una considerable mejora en la tasa de error de test respecto a los resultados obtenidos en las redes neuronales de la práctica anterior. Con el perceptrón simple y adaline la tasa de error rondaba o superaba el 30%, mientras que con el perceptrón multicapa la tasa de error se disminuye a un 3% en el problema real 1 y a un 20% en el problema real 2. Cambiando el umbral del error cuadrático medio es posible reducir las épocas de los problemas reales obteniendo todavía un porcentaje de error pequeño.

Tarea 3: predicción en problemas con más de dos clases

Aplicamos las mismas pruebas de la tarea 2 con la base de datos problema-real-3clases.txt. La diferencia en esta base de datos con las anteriores es que contiene más de dos clases que clasificar.

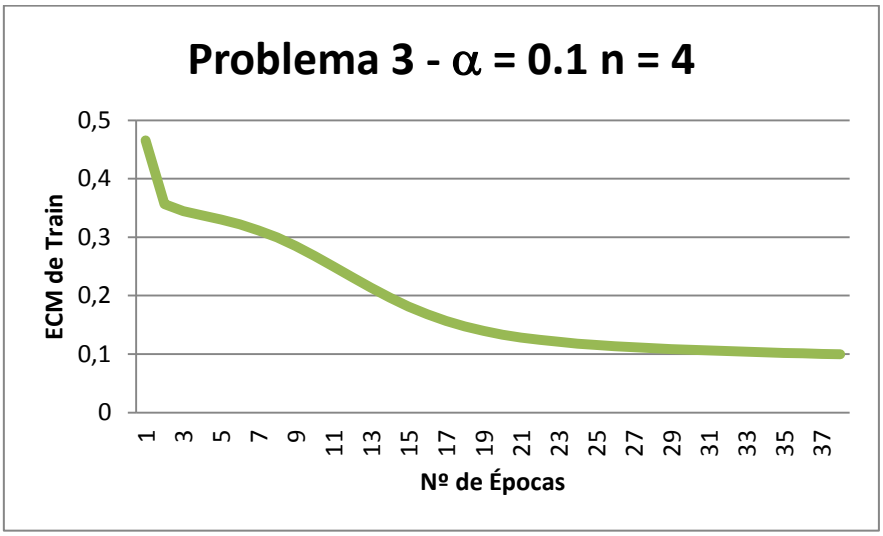
En este caso antes de realizar las pruebas ha sido necesario mezclar aleatoriamente los patrones dados, puesto que al estar ordenadas por clases las muestras de entrenamiento y test eran de clases distintas y la tasa de error era obviamente elevado.

Resultados

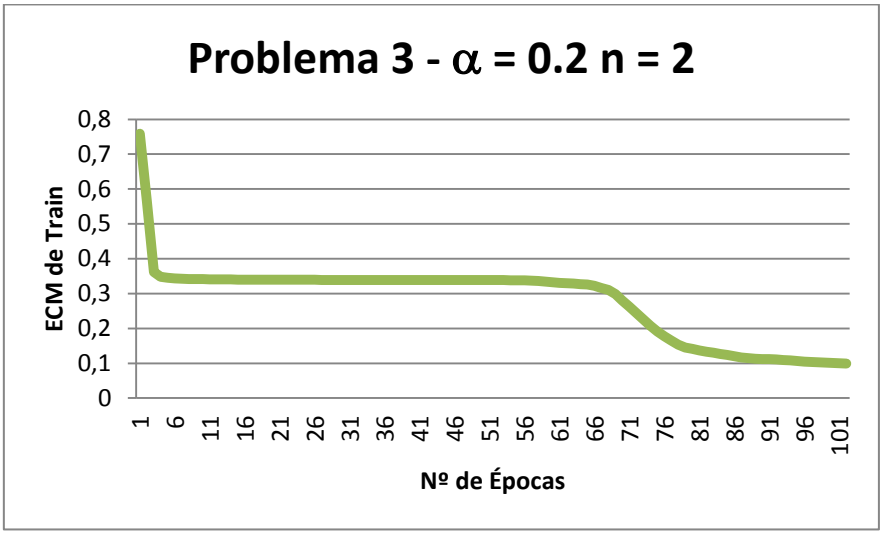
A continuación, se muestran los resultados obtenidos en tras realizar las pruebas. En el título de cada gráfica se indica el valor de la constante de aprendizaje (α) y el número de neuronas en la capa oculta (n). En las gráficas se representa el valor del error cuadrático medio (ECM) en cada época durante el entrenamiento. Sobre cada gráfica se indica la tasa de error en el test.

problema_real_3clases.txt

Tasa de error del Test: 4%



Tasa de error del Test: 4%



Análisis de los resultados

Con estas pruebas comprobamos que la red neuronal con el algoritmo de retro-propagación es capaz de clasificar más de dos clases. En este caso podemos observar que la tasa de error del test es muy pequeña de un 4%. Al igual que en el apartado anterior se observa que con un mayor número de neuronas en la capa oculta se alcanza en menos épocas el mínimo absoluto.

Tarea 4: predicción de un problema complejo

Aplicamos las mismas pruebas de la tarea 3.

Nuevamente, antes de realizar las pruebas ha sido necesario mezclar aleatoriamente los patrones dados. Además haremos una estadística de los datos de entrenamiento.

Resultados

A continuación, se muestran los resultados obtenidos tras realizar las pruebas. En el título de cada gráfica se indica el valor de la constante de aprendizaje (α) y el número de neuronas en la capa oculta (n). En las gráficas se representa el valor del error cuadrático medio (ECM) en cada época durante el entrenamiento. Sobre cada gráfica se indica la tasa de error en el test.

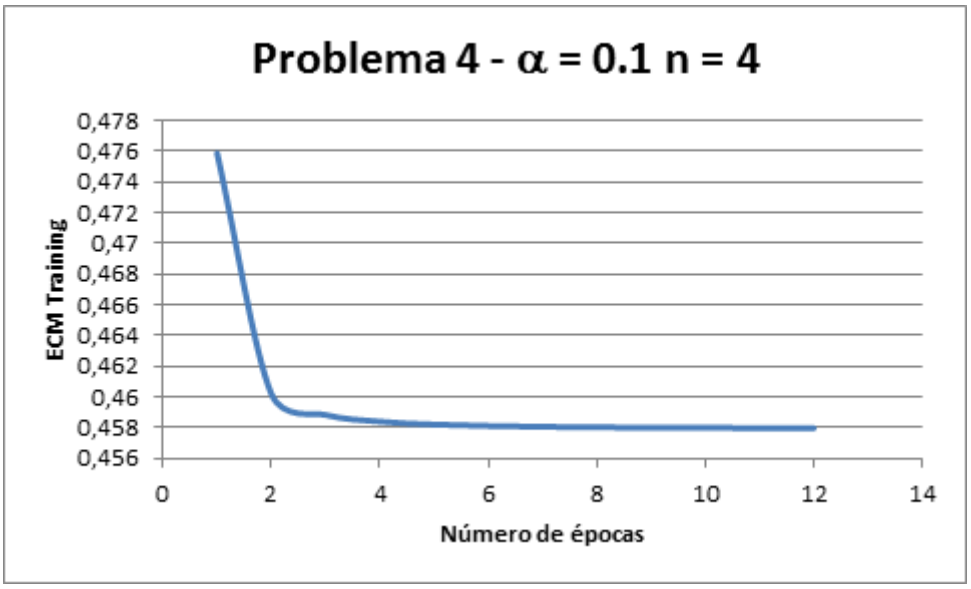
Estadística de los datos de entrenamiento:

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9
Media	4379.8283	0.0329	0.0330	28.9270	0.6613	0.0356	0.0351	0.0302	1615.8798
Desvío	2738.7189	0.0317	0.0303	28.8437	0.4619	0.0358	0.0254	0.0317	1776.9385

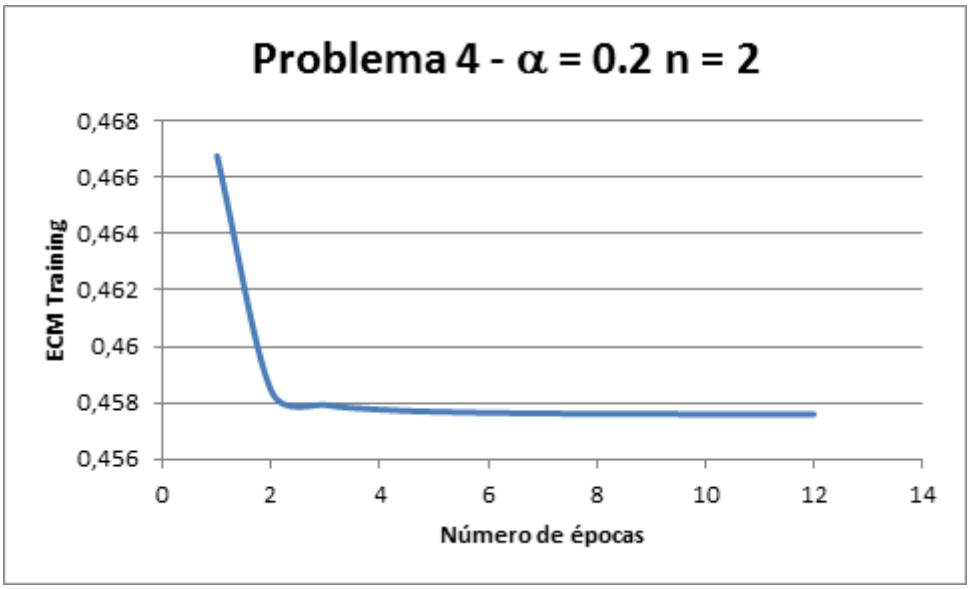
Tabla 1: Estadísticas del fichero problema_real_4.txt

Base: problema_real4.txt (no normalizado)

Tasa de error del Test: 32.6180%



Tasa de error del Test: 32.1888%



Análisis de los resultados

Los resultados comprueban lo que se ve en la tabla de estadísticas del fichero de entrada. El fichero de entrada no está normalizado y esto genera problemas en el sentido de que hay entradas que influyen directamente en la salida de manera más importante.

El desequilibro de las entradas genera pesos que influyen más que otros de manera completamente errónea y eso, al afectar todas las neuronas, se traduce en salidas incorrectas del modelo.

Tarea 5: normalización de los datos

Aplicamos las mismas pruebas de la tarea 4 y ahora debemos proceder una normalización de los datos con base en los datos de entrenamiento. Eso significa que los datos de test serán normalizados con base en estadísticas de la base de entrenamiento. Dichas estadísticas se pueden ver en la misma tabla 1 de la tarea 4.

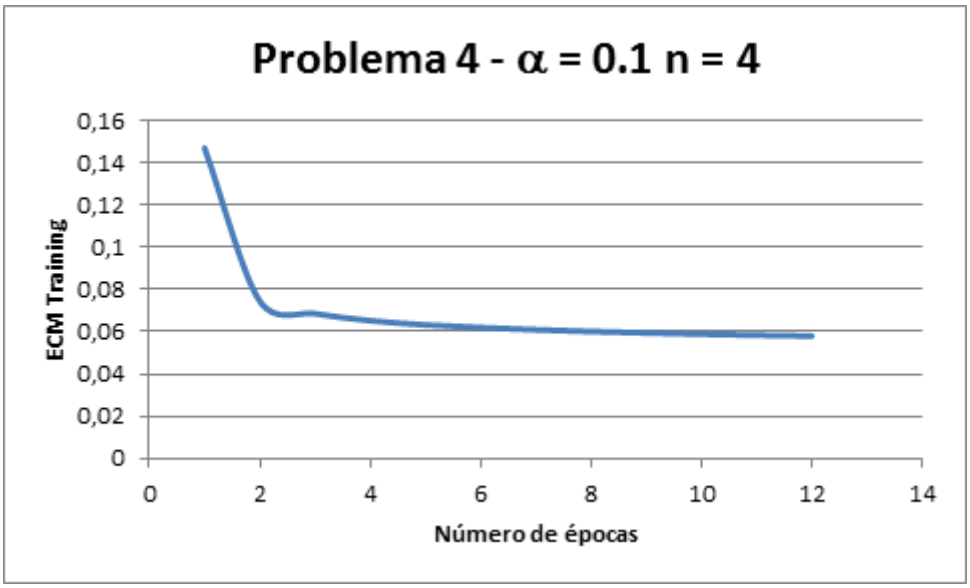
Nuevamente, antes de realizar las pruebas ha sido necesario mezclar aleatoriamente los patrones dados. La normalización se hace luego en el siguiente paso.

Resultados

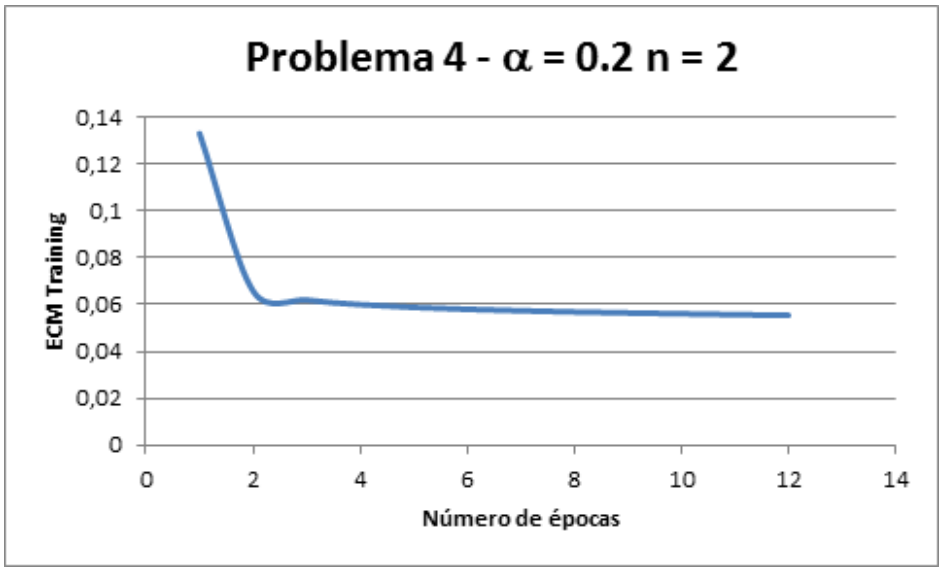
A continuación, se muestran los resultados obtenidos tras realizar las pruebas. En el título de cada gráfica se indica el valor de la constante de aprendizaje (α) y el número de neuronas en la capa oculta (n). En las gráficas se representa el valor del error cuadrático medio (ECM) en cada época durante el entrenamiento. Sobre cada gráfica se indica la tasa de error en el test.

Base: problema_real4.txt (normalizado)

Tasa de error del Test: 0.8583%



Tasa de error del Test: 3.0042%



Análisis de los resultados

El efecto normalización fue determinante y eso se ve en las gráficas de arriba. Los resultados contrastan de manera muy evidente y se ve en este apartado como las entradas normalizadas tienen fundamental importancia para que el cambio de pesos no se vaya a una dirección errónea.

Mirando las gráficas, se ve que con una tasa de aprendizaje más baja el test fue prácticamente perfecto con su porcentaje de errores en 0.8583%.

Tarea 6: predicción de datos no etiquetados

Debemos realizar muchas pruebas y analizar los modelos generados en la tarea 2 sobre el fichero “problema_real2.txt” al fin de buscar el mejor modelo posible para la predicción del fichero “problema_real2_no_etiquetados.txt”.

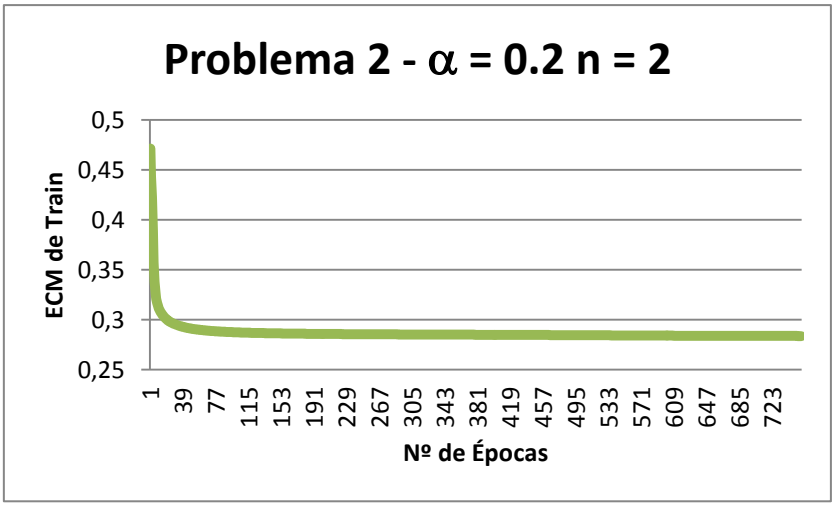
Resultados

A continuación, se muestra la gráfica del mejor modelo encontrado tras realizar las pruebas. En el título de la gráfica se indica el valor de la constante de aprendizaje (α) y el número de neuronas en la capa oculta (n). Se representa el valor del error cuadrático medio (ECM) en cada época durante el entrenamiento. Sobre la gráfica se indica la tasa de error en el test.

La predicción del fichero “problema_real2_no_etiquetados.txt” se encuentra en otro fichero llamado “predicciones_nnet.txt”.

Base: problema_real2.txt

Tasa de error del Test: 20.32085561497326%



Análisis de los resultados

Buscamos bajar la tasa de errores en test, pero no fue posible disminuir los 20%. Así que el mejor y el modelo responsable por las predicciones de esta tarea está basado en una tasa de aprendizaje de 0.2 y dos neuronas en la capa oculta.

Conclusiones

En esta práctica hemos aprendido a implementar una red neuronal multicapa con una sola capa oculta entrenada con el algoritmo de retropropagación. Hemos podido comprobar con las pruebas que el algoritmo de retropropagación utiliza el método por descenso de gradiente para minimizar el error cuadrático de la salida, balanceando de forma adecuada la memorización de los patrones de entrenamiento y la generalización de los patrones de test.

A diferencia de las redes neuronales analizadas en la práctica anterior, el perceptrón simple o adaline, el multicapa es capaz de resolver problemas que no son linealmente separables. El problema que encontramos en la red neuronal multicapa es que si se entrena de forma insuficiente la red, las salidas son imprecisas, o si llegamos a un mínimo local, como nos ocurre con la puerta lógica XOR. La solución es modificar la red cambiando los pesos iniciales, el número de neuronas, la constante de aprendizaje o cambiar el conjunto de patrones de entrenamiento y test.